L8: Entry 9 of 10 File: USPT Jan 7, 1997

DOCUMENT-IDENTIFIER: US 5592664 A
TITLE: Database server system with methods for alerting clients of occurrence of database server events of interest to the clients

Abstract Text (1):
An event alerter system for notifying one application or process of a change in a database. A database management system includes an event generator that defines events to be reported, such as a change in a particular field of the database. Each time an change occurs, the event generator notifies an event manager of the name of the event. Whenever an event dependent process indicates an interest in a change in a field in a particular record or records in a database, it transfers a command to the event manager identifying each such record and change. When the event dependent process issues such a command, it enters a wait state to process the occurrence of an event in either a synchronous or asynchronous mode. The event manager converts such a command into entries in an event table that identifies, for that and other event dependent processes, the list of events in which a process has an interest. Thereafter, each time the event generator signals a change, the event manager examines the event table to determine which, if any, event dependent processes have an interest in that particular change and only three event dependent processes that have requests for being notified of the change pending. Then the event dependent process can obtain a message containing the status of each event in which it has an interest.

Brief Summary Text (5):
Database management systems have become important tools recent years because they enable a quick and inexpensive retrieval of specific data from a large amount information in response to a number of criteria and facilitate the maintenance of that information. Database management systems have two basic components commonly called a "database" and a "database manager" that typically reside in a data processing system or a network of data processing systems. Databases include census information, airline schedules, stock prices and library catalogs to name a few examples and typically are stored in disk storage facilities. A database manager includes various software that enables the maintenance of and access more databases. It is the database manager that enables the addition, deletion and modification of data in the database and the transfer of data from the data base.

Brief Summary Text (8):
Although this invention can be implemented on a number of different database management systems, the invention and its background are most readily understood by describing them in terms of a relational database management system provided by the assignee of this invention, Interbase Software Corporation. A relational database as found in such a system contains data that users perceive as a collection of relations. Each such relation comprises a number of records and each record contains one or more fields. Each field has a name and contains data. In other database management systems phrases such as "table-row-column", "table-tuple-domain" or "file-record-field" are synonymous for the "relation-record-field" syntax used in the Interbase Software Corporation relational database management system.

Brief Summary Text (10):
Database management systems usually allow the construction of various applications or user programs that modify the database by adding or deleting records or by changing fields in the records or by adding fields to or deleting fields from the records, all on a selected basis. Other applications produce reports or evoke other responses based upon selected data in the database. For example, an application might generate a report listing the name and price of every record or a subset of selected records from the previously defined commodities or monitor databases.

Brief Summary Text (11):
It oftentimes is desirable to produce a report or evoke some action whenever a value in a particular field changes or values in a combination of fields change. A report, for example, might take the form of a message sent to a particular broker using the commodities database whenever the price changes for a particular commodity or subset of commodities. Evoking an action might take the form of initiating some particular control action whenever the output from a sensor, as recorded in that sensor's Sensor Value field, exceeds some threshold value.

Drawing Description Text (18):
FIG. 16 depicts further detailed steps the event manager uses during the declaration procedure as shown in FIGS. 13A and 13B and other procedures for determining the status of a particular interest block in the event table of FIG. 2;

Detailed Description Text (80):
In general terms, the event manager 24 in FIG. 1 responds to the first three statements by making appropriate entries in the event table 25. The event dependent process enters a wait state after it executes the third statement. The event manager issues an immediate notice that an event has occurred (step 106), so there is essentially no delay between the processing of the third statement and the fourth statement whereby the event dependent process receives the status of each of the named events in its event and result buffers.

Detailed Description Text (93):
The use of a trigger in the database as an event generator allows the database system to monitor events without any communications with event dependent processes. As will become more apparent later, the event manager 24 essentially passes each signal from the triggers through filters that each event dependent process defines. As these filters are located in a single table with the event manager 24, the event manager only notifies an event dependent process if that process has indicated a present interest in being notified and the event occurs. As a result, each event dependent process receives timely notification of events of interest with a minimal level of communications between the database management system and the process. This invention is particularly helpful in improving performance when the database management system is installed on a network. The use of preprocessors and asynchronous traps also enables simple interfaces to exist between data processing equipment at different nodes in both homogeneous and heterogeneous networks.

Detailed Description Text (96):
The operating system will assign to each event dependent process that accesses the event manager 24 a process identification that uniquely identifies that process in the event manager 24. Normally an event dependent process will constitute a single session for that process. In some situations, however, an event dependent process may have distinct sections that must be handled as separate processes. In accordance with this embodiment of the invention, such separate sections appear as separate sessions under a single process. In either case, the first time a process or session accesses the event manager, auxiliary procedures (see step 131 in FIG. 7) invoke an "EVENT.sub.--create.sub.-- session" operating sequence in the event manager 24 with a status vector address as an argument. As now described, when the event manager 24 completes the operating sequence in response to this command, the event table 25 will have a process block that includes the process identification assigned by the operating system and a session block assigned to the event dependent process, and the event dependent process will have received pointers to these process and session blocks.

Detailed Description Text (116):
As described with respect to FIGS. 5 and 7, the EVENT.sub.-- que command is an example of a command that declares an interest in an event. This command contains the location of a status vector, the session identification produced in accordance with FIG. 9A and 9B, a designation of the database and a list of the names of each event of interest to the event dependent process. The entry in the event list contains two components, namely: (1) a length byte and (2) an event name. The length byte identifies the number of characters in the event name. In the context of the commodities database, the auxiliary programs could declare an interest in soybeans and pork bellies by issuing a command in the form:

Detailed Description Text (143):
Referring again to FIG. 13B, if all the foregoing operations occur without any post flag being set, the event manager 24 does not use the posting process in step 253. In either case, the event manager 24 ends the sequence of FIG. 13B by relinquishing

exclusive use of the event table 25 in step 258 in accordance with the sequence of FIG. 12. Then the event manager 24 returns a status vector indicating successful operation and returns the request identification through the operating system to the event dependent process (step 259).

Detailed Description Text (149):
When the database manager 23 executes the commit statement (step 263), it makes permanent changes to the database and, with appropriate timing, issues an EVENT.sub.-- post command for each deferred work block in a post event deferred work block list. As shown in FIG. 18 the EVENT.sub.-- post command contains three basic arguments, namely: a status vector, an explicit identification of the database and an event name having the general form of a name length byte and the event name in succession as recorded in the corresponding deferred work block.

Detailed Description Text (151):
The event manager 24 then utilizes the database name in the EVENT.sub.-- post command to find a corresponding parent event block in the SRQ:EVENT BLOCKS self-relative queue in step 273 using the sequence of FIG. 14. If no such parent block is found, there are no pending requests for the database, so the event manager 24 branches in step 274 to relinquish exclusive use of the event table (step 275) and to return an appropriate status word that indicates a successful operation (step 276).

Detailed Description Text (152):
If the database manager finds the parent event block in step 273, it branches at step 274 to find an associated child event block that contains the event name recovered from the EVENT.sub.-- queue command. Step 277 again calls the procedure shown in FIG. 14. If no child event block is found, there is no interest in that event, so the event manager 24 branches from step 280 to step 275 to relinquish exclusive use of the event table 25 and return an appropriate status word.

Detailed Description Text (194):
The mechanism of this invention achieves all these objectives by utilizing a new "post event" trigger means that monitors any change in the named fields in the database and reports to the event manager independently of all other processes. The trigger is an example of an event generating means that produces a signal in the form of a post event command each time an event occurs. That is, the trigger can produce a post event command based on either a very simple condition or complex set of conditions. Moreover, a single generates a post event command each time that set of conditions is met anywhere in the database. Thus, if the commodities database of the example includes hundreds of commodities, the trigger in FIG. 3 will produce a post event command each time a change occurs in the price of any of those commodities, not just commodities of interest, although the post event command will identify the particular commodity that has been changed.

Current US Original Classification (1):
707/1

CLAIMS:

9. In a multi-user computer system, the system including a relational database having data fields storing values which change in response to clients in communication with a database server, a method for notifying at least one client of a change occurring in the database, the method comprising:

(a) defining preselected changes to the database to comprise a set of events which are of interest to the clients, said events including an event of a data value stored in a selected threshold;

(b) registering with said database server a request from at least one of said clients to be notified upon occurrence of a selected one of the events;

(c) receiving at said database server a request to change a data value stored in one of the data fields;

(d) determining by said database server if the request to change a data value specifies a change that is one of said preselected changes defining an event of interest; and

(e) if the request to change a data value specifies a change that is one of said preselected changes defining an event of interest, notifying all clients registered

with said database server of occurrence of the event of interest.

18. In a multi-user relational database management system, the system including a database server providing access to a database, the database server in communication with a plurality of clients, a method for notifying clients of particular events occurring in the database including information stored in data fields, the method comprising:

(a) defining a set of events indicating changes to the database which are of interest to at least one client, each of said at least one client requiring notification of occurrence of one of the events, and said set including an event which is defined to occur when a selected one of the data fields changes in value a preselected number of times;

(b) posting a transaction to the database server, said transaction specifying at least one of the events said at least one client requires notification of occurrence;

(c) upon occurrence of one of said events, determining by the database server which clients require notification of occurrence of said one of said events, said one of said events indicating a committed change in data in said database; and

(d) sending a notification from said database server to each said at least one client which requires notification of said one of said events which has occurred.

End of Result Set

L8: Entry 10 of 10 File: USPT Nov 28, 1995

DOCUMENT-IDENTIFIER: US 5471629 A
TITLE: Method of monitoring changes in an object-oriented database with tuned monitors

Brief Summary Text (28):
Another technique, access-oriented programming, is implemented in some object-oriented languages such as "LOOPS". A message to set values of instance variables is intercepted by means of a user-provided trigger procedure which may in turn set or display some other value (M. J. Stefik et al., "Integrating Access-Oriented Programming Into a Multiparadigm Environment," IEEE Software 3, 10, Jan. 1986, pp. 10-18). The trigger procedures are dynamically added and removed from running systems to avoid interfering with other system logic (K. Osterbye, "Active Objects: An Access Oriented Framework for Object-Oriented Languages," Journal of Object-Oriented Programming, Vol. 1, No. 2, June/July 1988, pp. 6-10).

Brief Summary Text (33):
When the initiating client requests that the transaction be committed, the system determines which monitored attributes may have been affected and then determines whether the values of any of said attributes have in fact changed. For each value which has changed, the system notifies any client which had requested monitoring of that attribute. The client which requests the commit is notified during the process of determining whether the values have changed. Other clients are notified after the update has been committed to the database.

Brief Summary Text (35):
"Notifying" a client means interrupting any task then being performed by the client and invoking a predesignated tracking procedure. The tracking procedure is an application program procedure that is called by the database system when a result change has been detected. Thus the tracker is part of the application program and is ordinarily written in the same language as that program. The database system does not send data to the tracker when a result change has occurred; instead, the tracker can freely access the database to retrieve the current result of the monitored query. Thus, change detection (invoking the tracker) is distinguished from retrieving the new result.

Brief Summary Text (45):
In another aspect of the invention, if a client requests notification of any changes which have occurred in an attribute then being monitored for said client, the system determines whether a predetermined criterion respecting a change in the monitored attribute has been satisfied and, if the criterion has been satisfied, the system notifies said client accordingly. This notification invokes the tracker of that client.

Brief Summary Text (47):
In the case of the change significance parameter, the criterion is a minimum change value in the monitored attribute. Whether the criterion has been satisfied is determined by determining whether the monitored attribute may have been affected by the update transaction and, if so, determining whether the value of the attribute has changed by an amount which exceeds the minimum change value. Whether the attribute has changed, and if so by how much, are determined by computing an updated value for the attribute and comparing the updated value with the value in the attribute value record. Thus, the client is notified only of changes that exceed a predetermined significance level.

Brief Summary Text (48):
In the case of the tracking delay time parameter, the criterion is a minimum time

interval. Whether the criterion has been satisfied is determined by determining whether the monitored attribute may have been affected by said transaction and, if so, determining whether an amount of time that exceeds the minimum time interval has elapsed subsequent to a previous event. If the minimum time interval has elapsed, the system goes on to determine whether the value of the attribute has changed by computing an updated value for the attribute and comparing the updated value with the value in the attribute value record. The previous event typically is a previous change in the value of the monitored attribute. Thus, the client is not notified of changes more often than once in a defined interval of time.

Detailed Description Text (2):
As shown in the drawings for purposes of illustration, the invention is embodied in a novel method of monitoring an attribute of a database object wherein a client program is notified of a change only if a selected monitor criterion is satisfied. There has been a need for an efficient way to control monitoring of database objects so that a client is notified neither more nor less often than necessary.

Detailed Description Text (3):
A method of monitoring an object according to the present invention includes keeping records of monitor requests, interdependence relationships between monitored attributes and other attributes, values of monitored attributes, and update transactions. If a predetermined criterion respecting a monitored attribute is satisfied, the client is notified. The criterion may be any one or more of four tuning parameters which include a change significance parameter, a tracking delay time parameter, a nervousness parameter, and a synchronous initiation parameter. Monitoring database objects by means of these criteria leads to improved efficiency of the database system and optimal results from each of a plurality of client programs.

Detailed Description Text (16):
Notifying a client preferably comprises interrupting any task then being performed by the client and invoking a predesignated tracking procedure, for example by sending the notification to a portion of the client program referred to as a "monitor server". If the client being notified is the one which called the Check Monitors procedure (usually this will be the client which requests that updates be committed), this is done when a change in the monitored attribute value is detected (block 309 of FIG. 3).

Detailed Description Text (17):
If the client being notified is not the client which called the Check Monitor procedure, a determination is made, for example by that client's monitor server, whether the client is in an interruptible state (block 212 of FIG. 2). If the client is in an interruptible state, the monitor server invokes the tracker (block 213). If the client is not in an interruptible state, the monitor server waits (block 214) and keeps a record of the notification either until the client enters an interruptible state at which time the monitor server interrupts the client or until the client requests notification.

Detailed Description Text (35):
The change significance parameter provides one way to reduce the reactivity of a monitor activation. This parameter has the effect of making the tracker react only to "significant" changes in the value of the monitored attribute. The client defines what is a "significant" change by specifying a minimum value for the change significance parameter. This minimum is specified either as an absolute difference (for example, ten dollars) or as a relative difference (for example, a change of 10%). For a query that returns a set of values, the significance test is preferably applied to every value in the set such that if any value changes by an amount that exceeds the minimum change value, the client is notified.

Detailed Description Text (51):
The synchronous notification requirement will now be discussed with reference to FIG. 7, which illustrates the "commit" procedure as represented by the block 208 in FIG. 2. If the criterion includes synchronous notification, as indicated by a block 217, the actual commit operation 219 is delayed, as indicated by a block 221, until the requesting client has been notified that the monitored attribute has been changed. Typically the commit operation need not be delayed any longer; that is, there is no need to await completion of any procedure performed by the notified client as a result of the notification. Synchronous notification only pertains to notification of clients other than the one that initiates the update.

Detailed Description Text (53):

It will be apparent that any such changes will be changes which have not yet been committed to the database. Accordingly, notification of such changes is not sent to any client other than the one which called the Check Monitors procedure, and synchronous notification would not be applicable. However, a record of such changes is kept (block 305) for later notification of other clients and the Function Change table is cleared (block 307). If there are no further changes in the monitored value, then the notifications are sent to the other clients at commit time (as indicated by blocks 209 and 215 in FIG. 2 and 310 in FIG. 3). If there are further changes, then such changes will be detected at commit time and appropriate notifications will be sent. Thus, clients other than the client which invokes the Check Monitor procedure are assured that changed data either have been committed to the database at the time the clients are notified of the changes or, if synchronous notification is specified, will be committed synchronously with the notification.

Detailed Description Text (60):
At time T8 the update client decides to commit whatever updates it has initiated. In response, the system determines whether the specified monitoring criterion has been satisfied. If the criterion has been satisfied, the update client's monitor server is notified. The monitor server interrupts the client (time T9) and invokes the tracking procedure which the update client had designated earlier at time T3, when it first requested monitoring. The invoked procedure ends at time T10 and the system continues by clearing the Function Change table, committing the data, and notifying the remote client's monitor server, and committing the data. Of course, if the remote client had specified synchronous notification, the actual commit would be delayed until after the remote client had been notified.

Detailed Description Text (68):
An example of how communications might pass among client programs and the database system is shown in FIG. 10. For purposes of discussion all software and data are depicted as residing in main memory, but it will be apparent that in a real system some of the software and data are actually resident in mass storage and are called into main memory as needed. Also, certain communication channels are depicted as physical lines of communication but may actually exist in some other form such as instructions which pass information back and forth among different computer programs or parts of a program.

Detailed Description Text (69):
The memory 703 of the main computer 701 contains the database system (software and data) 715, a client program 717, a monitor server 719, and other programs 721. The client program 717 can issue commands to the database through a communication channel 723. The monitor server 719 can interrupt the client program and invoke a predesignated procedure through a communication channel 725. The monitor server 719 receives notifications from the database 715 through a communication channel 727.

Detailed Description Text (79):
Failures of parts are reported by field engineers. These failure reports are used to calculate failure rates. Each such rate is represented in the database as an entity of type Failure. A relationship PartFailure 808 between parts and failure rates, and a relationship ProductFailure 809 between products and failure rates, together define which parts are failing in which products and with which failure rates.

Detailed Description Text (84):
A database update program, which for convenience will be referred to as the "failure rate reporter" is run at fixed time intervals. It is run by customers to report part failures and to store this information in the database. The failure rate reporter also calculates current failure rates from the reports received for the various parts during the current time interval. The failure rate reporter is independent of the service and producer monitors.

Detailed Description Text (90):
In a conventional database system, a client such as the service and producing departments would have to regularly query the database for every situation they might regard as important. With database monitors the system actively notifies the clients. The tuning parameters optimize the performance of the system by assuring that each client is notified neither too often nor not often enough.

Detailed Description Text (93):
As has been discussed above, database monitors use various system tables, some of which are updated very seldom. For example, when a monitor is defined (compiled) the system

analyzes the query to be monitored and stores dependency information in system tables. This is done only once for each monitored query. The dependency information is traversed when transactions screen out non-monitored updates. The dependency information is thus updated rarely but accessed intensively and is therefore a good candidate for materialization as view objects. These view objects should be organized for quick traversal of the screening tests.

Detailed Description Text (98):
View materialization may be considered as a way to implement an "identity connection". The identity connection was introduced by Wiederhold et al., "Modeling Asynchrony in Distributed Databases", Third International Conference on Data Engineering, Los Angeles, Calif., Feb. 3-5, 1987, pp. 246-250. The identity connection defines replicated attributes of relations and is regularly updated. A derivation formula can transform the desired attribute. Wiederhold, "Connections", in Wiederhold et al., Managing Objects in a Relational Framework, Stanford Computer Science Report #STAN-CS-89-1245, 1989.

Detailed Description Text (103):
An "event" can reestablish a connection. This is implemented with a database monitor by recording the time of the event and then monitoring the recorded data.

Current US Original Classification (1):
707/201

CLAIMS:

6. A method as in claim 1 wherein:

the criterion comprises a change in the value of said monitored attribute;

determining whether the criterion has been satisfied comprises determining whether said monitored attribute may have been affected by said transaction, and if said attribute may have been affected, determining whether the value of the attribute has changed by computing an updated value for the attribute and comparing the updated value with the value in the attribute value record; and

notifying a client comprises notifying the client synchronously with storing the update transaction in the database.

7. A method as in claim 1 wherein:

the criterion comprises a minimum change value in said monitored attribute;

determining whether the criterion has been satisfied comprises determining whether said monitored attribute may have been affected by said transaction, and if said attribute may have been affected, determining whether the value of the attribute has changed by an amount which exceeds the minimum change value by computing an updated value for the attribute and comparing the updated value with the value in the attribute value record; and

notifying a client comprises notifying the client synchronously with storing the update transaction in the database.

8. In a computer database system, a method of monitoring an object in a database in response to a request from any of a plurality of client programs, the method comprising the following steps carried out by the system:

keeping a record of any request from a client to monitor an attribute of the object;

keeping a record indicating any interdependence relationships between the attribute being monitored and other attributes;

keeping a record of a value of each attribute being monitored by accessing said interdependence relationship record;

during a database update session, keeping a record of any database update transactions initiated by a client; and

if a client requests notification of any changes which have occurred in an attribute

then being monitored for said client, determining whether a predetermined criterion pertaining to a change in the monitored attribute has been satisfied and, if the criterion has been satisfied, notifying said client that the criterion has been satisfied;

wherein said criterion comprises either of:

a) a minimum change value in said monitored attribute and determining whether the criterion has been satisfied comprises:

determining whether said monitored attribute may have been affected by said transaction; and

if said attribute may have been affected, determining whether the value of the attribute has changed by an amount which exceeds a minimum change value by computing an updated value for the attribute and comparing the updated value with the value in the attribute value record; or

b) a minimum time interval and determining whether said criterion has been satisfied comprises:

determining whether said monitored attribute may have been affected by said transaction;

if said attribute may have been affected, determining whether an amount of time that exceeds a minimum time interval has elapsed subsequent to a previous event;

if the minimum time interval has elapsed, determining whether the value of the attribute has changed by computing an updated value for the attribute and comparing the updated value with the value in the attribute value record.

WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L6 and (notif$ near client)

Display: 50 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences | Cases

Search History

DATE: Thursday, May 29, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L8	L6 and (notif$ near client)	10	L8
L7	L6 nd (notif$ near client)	71175	L7
L6	L5 and (monitor$ near system)	51	L6
L5	L4 and (status or report)	185	L5
L4	L3 and (agent or software)	240	L4
L3	L1 and (monitor$ near (event$ or transact$))	274	L3
L2	L1 and event$	6913	L2
L1	((707/$)!.CCLS.)	15744	L1

END OF SEARCH HISTORY

WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L6 and (notif$ near client)

Display: 50 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ Hit List ◉ Hit Count ○ Side by Side ○ Image

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences | Cases

Search History

DATE: Thursday, May 29, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L8	L6 and (notif$ near client)	10	L8
L7	L6 nd (notif$ near client)	71175	L7
L6	L5 and (monitor$ near system)	51	L6
L5	L4 and (status or report)	185	L5
L4	L3 and (agent or software)	240	L4
L3	L1 and (monitor$ near (event$ or transact$))	274	L3
L2	L1 and event$	6913	L2
L1	((707/$)!.CCLS.)	15744	L1

END OF SEARCH HISTORY